Real Time Signal Processing Using FPGA

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE

BACHELORS OF ENGINEERING

By:

GURKIRAN SINGH BHATIA

ROHAN NALAVADE

HARSH RATHORE

Under the Guidance of

Mrs. NEERU PATHAK

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATIONS ENGINEERING

THADOMAL SHAHANI ENGINEERING COLLEGE

T.P.S.III, 32nd ROAD, BANDRA (W), MUMBAI- 400050

2017-2018

Project Report Approval

This project report entitled **REAL TIME SIGNAL PROCESSING USING FPGA** is approved for the degree of **BACHELOR OF ENGINEERING**

Examiners

1. _____
2. _____

Supervisor

1. _____

Head of the Institution

1. _____

Date:

Place:

Acknowledgement

We take this opportunity in representing the report on our project "**REAL TIME SIGNAL PROCESSING USING FPGA**". Kind deeds are simple and easy to do but their perfection is quite difficult. We would hereby like to acknowledge all those who have made this project possible in given span of time.

Our heart-felt thanks to Guide **Mrs. NEERU PATHAK** for her unflinching support. She has patiently, throughout all the obstacles, guided our creative efforts with gentleness and wisdom and has helped us every time when needed. She has always been a source of inspiration to all of us.

We would also like to thank our entire Electronics and Telecommunication Department and Humanities Department Staff, which has instilled in us the self-discipline that was so necessary in building the project.

Declaration

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have neither misinterpreted nor fabricated nor falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been cited or from whom proper permission has not been taken when needed.

_____

Gurkiran Singh Bhatia

_____

Rohan Sudhir Nalavade

_____

Harsh Kamlesh Rathore

Abstract

Signal processing ranks among the most demanding applications of digital design concepts. It is a mature technology domain wherein the demands for enhanced performance and reduced resource utilization have risen exponentially over the years. Field Programmable Gate Array (FPGA) design technology has becoming the preferred platform for evaluating and implementing signal processing algorithms. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an application specific integrated circuit (ASIC) for moderate volume applications, and more flexibility than the alternate approaches. Since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement different functionality if required. This describes an approach to the implementation of digital filter based on field programmable gate arrays (FPGAs) which is flexible and provides performance comparable or superior to traditional approaches, lowpower, area-efficient re-configurable digital signal processing architecture that is tailored for the realization of arbitrary response Finite impulse response (FIR) filters.

Contents

LIST OF FIGURES

1. Introduction

Recent advances in FPGA technology have enabled these devices to be applied to a variety of applications traditionally reserved for ASICs. FPGAs are well suited to datapath designs, such as those encountered in digital filtering applications. The density of the new programmable devices is such that a nontrivial number of arithmetic operations such as those encountered in digital filtering may be implemented on a single device. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches. In particular, multiple multiply-accumulate (MAC) units may be implemented on a single FPGA, which provides comparable performance to general-purpose architectures which have a single MAC unit. Further, since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement alternate filtering operations, such as lattice filters and gradient-based adaptive filters, or entirely different.

FPGAs are well suited for the implementation of fixed-point digital signal processing algorithms. The advantages of DSP on FPGAs are primarily related to the additional flexibility provided by FPGA reconfigurability. Not only can high-performance systems be implemented relatively inexpensively, but the design and test cycle can be completed rapidly due to the elimination of the integrated circuit fabrication delays. The new approach also allows adapting the functions to account for unforeseen requirements.

FIR FILTER ADD/SHIFT IMPLEMENTATION

In binary arithmetic, multiplication by a power-of-two is simply a shift operation. Implementation of systems with multiplications may be simplified by using only a limited number of power-of-two terms, so that only a small number of shift and add operations are required. An FIR filter tap as shown in can be implemented in two array columns of Xilinx series FPGAs. Because of the high degree of spatial and temporal locality, most signal

routing delays are not critical, as they are with typical high performance FPGA designs. Each of the bit slices for the tap requires two combinational logic blocks (CLBs) in the array for implementation.

This board has the following features, relevant to the presented implementation :

1) Spartan6 FPGA with 9152 logic cells (XC6SLX9),

2) 100MHz crystal oscillator,

3) Expansion connector with 32 I/O pins,

4) Flash memory(576kb) for bit stream storage,

5) USB port for FPGA configuration and memory programming.

The scheme for the implementation of FIR filter to FPGAs. The main components of the implemented circuit are as follows:

A. Memory: Prepared for storage of past position data of bunches. z-1 means 1-turn delay .

B. Adder: Adder is made to reduce the number of stages and is a key for stable operation of the FPGA used in the board. This reduction of the stage is effective to avoid errors by clock skew in the FPGA and to reduce the power consumption and a circuit area(or number of gates) on FPGA.

C. Multiplier: Build-in multipliers are used to fulfil the requirements of high-speed operation; therefore this number of build-in multipliers is one of the constraints to the number of taps of FIR filter.

D. Shift-Register: It is used for additional delay for adjust latency to one or two revolution period.

2. Review Literature

A Filter is frequency selective network, which is used to modify an input signal in order to facilitate further processing. Basically there are two types of filters-analog and digital. Digital Filters are widely used in different areas, because Digital filters have the potential to attain much better signal to noise ratio than analog filters. The digital filter performs noiseless mathematical operations at each intermediate step in the transform and their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters.

Digital filters operate on numbers opposite to analog filters, which operates on voltages. The basic operation of digital filter is to take a sequence of input numbers and compute a different sequence of output numbers. There exists a range of different digital filters. FIR and IIR filters are the two common filter forms. A drawback of IIR filters is that the closed-form IIR designs are preliminary limited to low pass, band pass, and high pass filters, etc. secondly FIR filters can have precise linear phase. Also, in the case of FIR filters, closed-form design equations do not exist and the design problem for FIR filters is much more under control than the IIR design problem. A FIR filter is a filter structure that can be used to implement almost any sort of frequency response digitally. It is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. The architecture of FIR filter is shown in Fig.

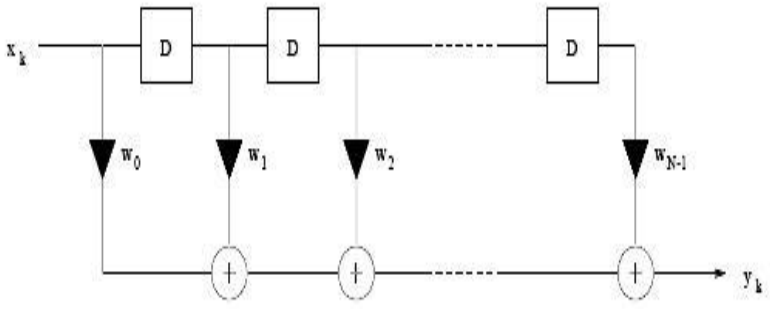


Fig 2 Basic FIR Filter Structure

From this structure the transfer function of canonic form of the filter can be easily described in Z-domain as:

$$H(Z)=w_0+w_1z^{-1}+w_2z^{-2}+------+w_{M-1}Z^{L-1}$$

In addition the advantages of FIR filter generally we use raised cosine pulse. The rectangular pulse occupies a large bandwidth so an alternative to rectangular pulse is a sinc pulse, which reduces the bandwidth and Inter Symbol Interference. The rectangular pulse is passed through the Root Cosine Filter a set of FIR filters to pulse shape the pulses to sinc. If very high sampling rates are required, full parallel hardware must be used. Such filters can be implemented on FPGAS using combinations of the general-purpose logic fabric, on-board RAM and embedded arithmetic hardware. Full-parallel filters cannot share hardware over multiple clock cycles and so tend to occupy large amounts of resource. Hence, efficient implementation of such filters is important to minimize hardware requirement.

3. Aim of the Project

The aim of the project involves the development of FIR filters on Field programmable gate array (FPGAs) using IP cores.

FIR filter has been designed and realized by FPGA for filtering the digital signal. The implementation of FIR filter on a Xilinx XC65LX9 FPGA is considered and the coefficients are computed using Online FDA tool. The model is capable of performing filtering operations like low pass, high pass, and band pass based on the coefficients selected. The most basic functions required for nearly any signal processor include addition, multiplication and delays. The analog input given is converted to a 12 bit digital data. IP Core has been used to filter the input data. The design is coded through Verilog (Hardware Descriptive Language). To verify the designed outputs simulation, compilation and synthesis have been done. To test the correctness of the design the observed output is compared with the calculated output results from MATLAB implementation that confirms the effectiveness of the design.

Computer-aided design tools are used to both simulate VERILOG design and to synthesize the design to actual hardware. The Xilinx Floating-Point core is a function inbuilt in IP cores provides designers with the means to perform floating-point arithmetic on an FPGA. The core can be customized to allow optimization for operation, word length, latency, and interface

The designing of an FIR filter in VERILOG with MATLAB (for the generation of coefficients of filter) and programming it onto an FPGA is done.

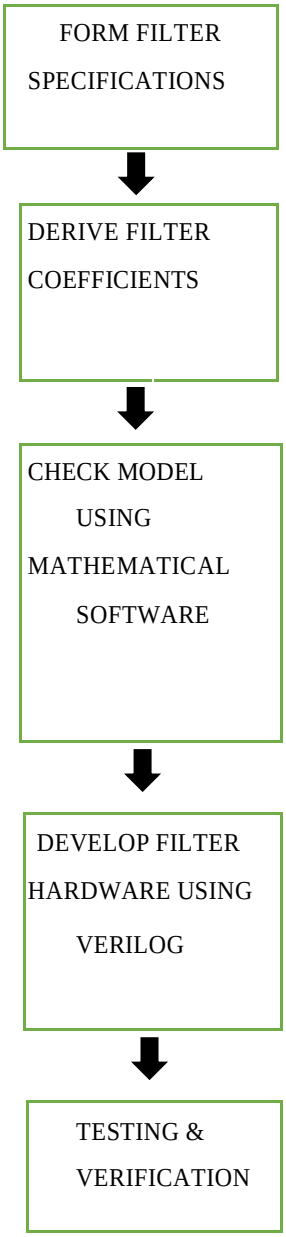
I. Implementation of project onto an FPGA (including hardware and software parts) VERILOG, MATLAB and basic digital filter concepts are used.

II. FIR Filters and its Characteristics FIR filters are digital filters with finite impulse response. They are also known as nonrecursive digital filters as they do not have the feedback (a recursive part of a filter).

III. Finite Impulse Response (FIR) filters are defined by scaled and time-delayed versions of the filter input signal only, as given by the following difference equation: y[n] =b0 x[n] +b1 x[n-1] +b2 x[n-2]..., n=0,1,2 …….. (1) where the input and output y[n]=0

IV. for n < 0.

Fig 3. Block diagram of a Basic Design Flow



4. DESIGN AND IMPLEMENTATION

4.1 Block Diagram

Block Diagram of the System which is implemented inside the FPGA is given below.

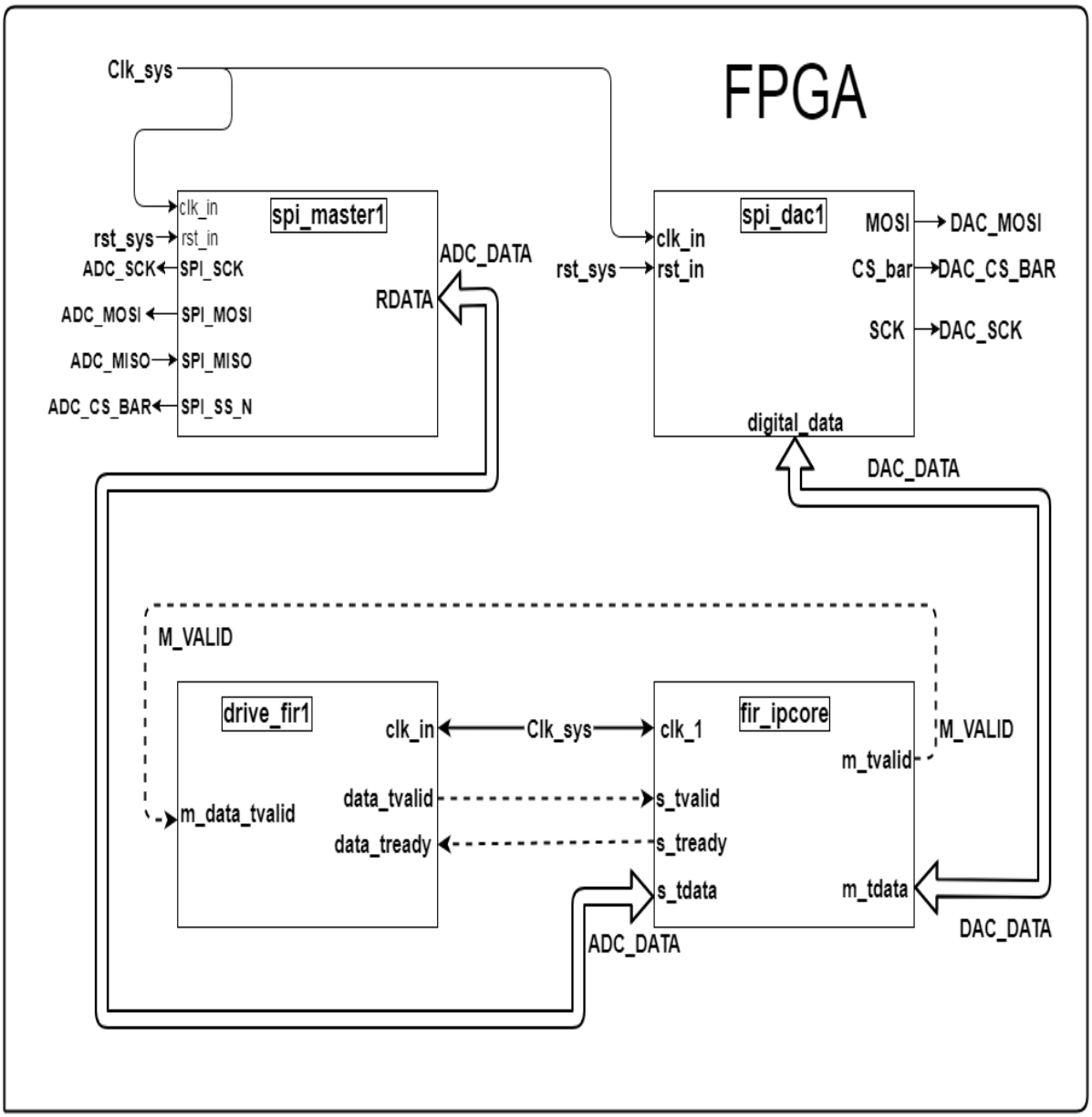


Fig 4 Block Diagram of the System

4.2 Working

The ‘clk_sys’ is the system clock of the FPGA given to the four blocks.Spi-master1 communicates with the external ADC.The data given to the ADC from the function generator is converted into a digital signal. This is then given to the FPGA. In the FPGA, the module spi_master1 will accept the input digital signal in a serial manner. The spi_master1 will then convert the serial input digital signal into a 12 bit parallel data referred as RDATA inside the spi_master1.

This data is then transferred to the fir_ipcore and is referred to as ADC_DATA. The fir_ipcore will accept the data and will do the filtering depending upon what kind of coefficients are loaded into the IP Core.

The fir_ipcore accepts the data when it is not processing any data. The decision of when the fir_ipcore should accept the data is made by the drive_fir1 block. The drive_fir1 block knows the rate at which the samples are coming from the spi_master1 and accordingly sends the data_tvalid signal which indicates to the fir_ipcore that a valid input has now come at s_tdata.

After processing the signal, the fir_ipcore sends the data to spi_dac1. The data is referred to as digital_data in the spi_dac1. Here the data is converted into serial form. spi_dac1 communicates with the external DAC. It transmits the serial data referred as DAC_MOSI. This data goes to the external DAC which converts this digital data into analog signal. This analog signal is then observed on the CRO.

5. COMPONENTS

5.1 Analog to Digital Converter

• 12-bit resolution

• Analog inputs programmable as single-ended or pseudo-differential pairs

• On-chip sample and hold

• SPI serial interface (modes 0,0 and 1,1)

• Single supply operation: 2.7V-5.5V

• 100 ksps max. sampling rate at VDD = 5V

• 50 ksps max. sampling rate at VDD = 2.7V

• Low power CMOS technology - 500 nA typical standby current, 5 µA max. - 550 µA max. active current at 5V

• Industrial temp range: -40°C to +85°C

• 8-pin MSOP, PDIP, SOIC and TSSOP packages

5.1.1 Description

The Microchip Technology Inc. MCP3202 is a successive approximation 12-bit Analog-to-Digital (A/D) Converter with on-board sample and hold circuitry. The MCP3202 is programmable to provide a single pseudodifferential input pair or dual single-ended inputs. Differential Nonlinearity (DNL) is specified at ±1 LSB, and Integral Nonlinearity (INL) is offered in ±1 LSB (MCP3202-B) and ±2 LSB (MCP3202-C) versions. Communication with the device is done using a simple serial interface compatible with the SPI protocol. The device is capable of conversion rates of up to 100 ksps at 5V and 50 ksps at 2.7V. The MCP3202 device operates over a broad voltage range (2.7V-5.5V). Lowcurrent design permits operation with typical standby and active currents of only 500 nA and 375 µA,

respectively. The MCP3202 is offered in 8-pin MSOP, PDIP, TSSOP and 150 mil SOIC packages.

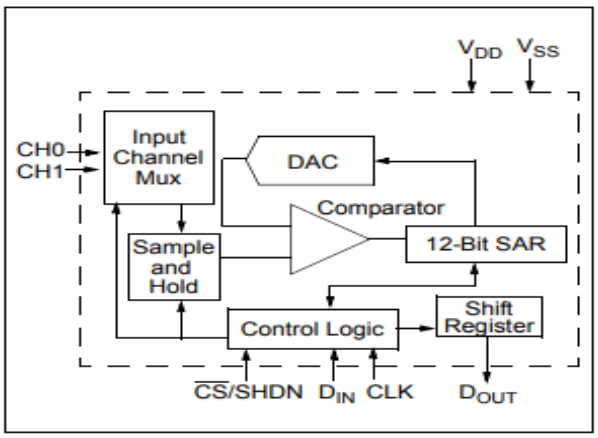


Fig 5.1.1 Functional Block Diagram of ADC MCP3202

5.1.2 Pin Description

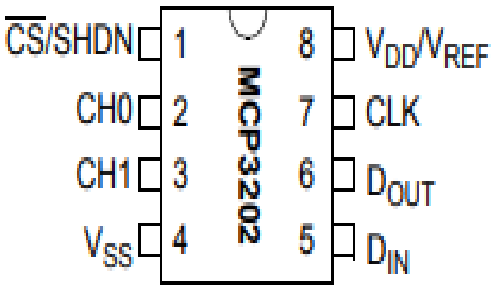


Fig 5.1.2 Pin Description of MCP3202

1. CH0/CH1

Analog inputs for channels 0 and 1 respectively. These channels can programmed to be used as two independent channels in single ended-mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN-. See Section 5.0 for information on programming the channel configuration.

2. Chip Select/Shutdown (CS/SHDN)

The CS/SHDN pin is used to initiate communication with the device when pulled low and will end a conversion and put the device in low power standby when pulled high. The CS/SHDN pin must be pulled high between conversions.

3. Serial Clock (CLK)

The SPI clock pin is used to initiate a conversion and to clock out each bit of the conversion as it takes place. See Section 6.2 for constraints on clock speed.

4. Serial Data Input (DIN)

The SPI port serial data input pin is used to clock in input channel configuration data.

5. Serial Data Output (DOUT)

The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

5.1.3 Serial Communication

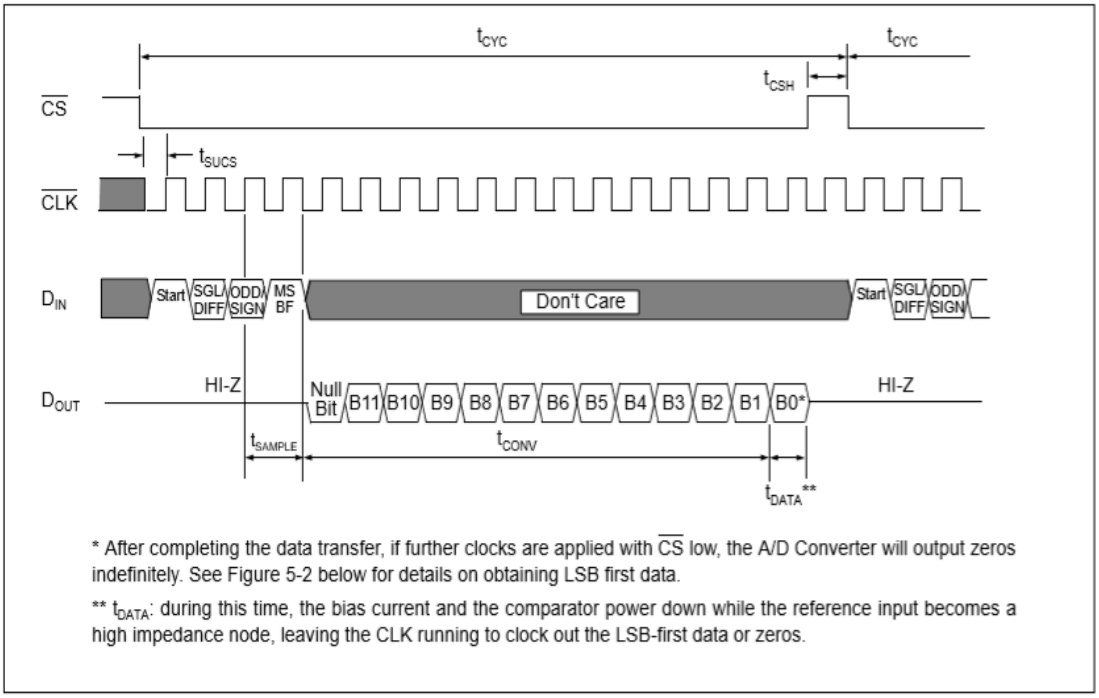
Communication with the MCP3202 is done using a standard SPI-compatible serial interface. Initiating communication with the device is done by bringing the CS line low. See Figure 5-1. If the device was powered up with the CS pin low, it must be brought high and back low to initiate communication. The first clock received with CS low and DIN high will constitute a start bit. The SGL/DIFF bit and the ODD/SIGN bit follow the start bit and are used to select the input channel configuration. The SGL/DIFF is used to select single ended or psuedo-differential mode. The ODD/SIGN bit selects which channel is used in single ended mode, and is used to determine polarity in pseudo-differential mode. Following the ODD/SIGN bit, the MSBF bit is transmitted to and is used to enable the LSB first format for the device. If the MSBF bit is high, then the data will come from the device in MSB first format and any further clocks with CS low will cause the device to output zeros. If the MSBF bit is low, then the device will output the converted word LSB first after the word has been transmitted in the MSB first format. The device will begin to sample the analog input on the second rising edge of the clock, after the start bit has been received. The sample period will end on the falling edge of the third clock following the start bit.

On the falling edge of the clock for the MSBF bit, the device will output a low null bit. The next sequential 12 clocks will output the result of the conversion with MSB first as shown in Figure 1. Data is always output from the device on the falling edge of the clock. If all 12 data bits have been transmitted and the device continues to receive clocks while the CS is held low, (and MSBF = 1).If more clocks are provided to the device while CS is still low (after the LSB first data has been transmitted), the device will clock out zeros indefinitely.

If necessary, it is possible to bring CS low and clock in leading zeros on the DIN line before the start bit. This is often done when dealing with microcontroller-based SPI ports that must send 8 bits at a time.

	Config Bits		Channel Selection		GND
	Sgl/ Diff	Odd/ sign	0	1	
Single Ended Mode	1	0	+	—	-
	1	1	—	+	-
Pseudo-DifferentiaL Mode	0	0	IN+	IN-	
	0	1	IN-	IN+	

Fig 5.1.3 Modes of Working of ADC



* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the A/D Converter will output zeros indefinitely. See Figure 5-2 below for details on obtaining LSB first data.

** t_{DATA}: during this time, the bias current and the comparator power down while the reference input becomes a high impedance node, leaving the CLK running to clock out the LSB-first data or zeros.

Fig 5.1.4 Timing Diagram of the ADC

5.2 DAC MCP4921

5.2.1 Features

• 12-Bit Resolution

• ±0.2 LSB DNL (typ)

• ±2 LSB INL (typ)

• Single or Dual Channel

• Rail-to-Rail Output

• SPI™ Interface with 20 MHz Clock Support

• Simultaneous Latching of the Dual DACs w/LDAC

• Fast Settling Time of 4.5 µs

• Selectable Unity or 2x Gain Output

• 450 kHz Multiplier Mode

• External VREF Input

• 2.7V to 5.5V Single-Supply Operation

• Extended Temperature Range: -40°C to +125°C

5.2.2 Description

The Microchip Technology Inc. MCP492X are 2.7 – 5.5V, low-power, low DNL, 12-Bit Digital-to-Analog Converters (DACs) with optional 2x buffered output and SPI interface. The MCP492X are DACs that provide high accuracy and low noise performance for industrial applications where calibration or compensation of signals (such as temperature, pressure and humidity) are required. The MCP492X are available in the extended temperature range and PDIP, SOIC, MSOP and TSSOP packages. The MCP492X devices utilize a resistive string architecture, with its inherent advantages of low DNL error, low ratio metric temperature coefficient and fast settling time. These devices are specified over the

extended temperature range. The MCP492X include double buffered inputs, allowing simultaneous updates using the LDAC pin. These devices also incorporate a Power-On Reset (POR) circuit to ensure reliable power-up.

Block Diagram

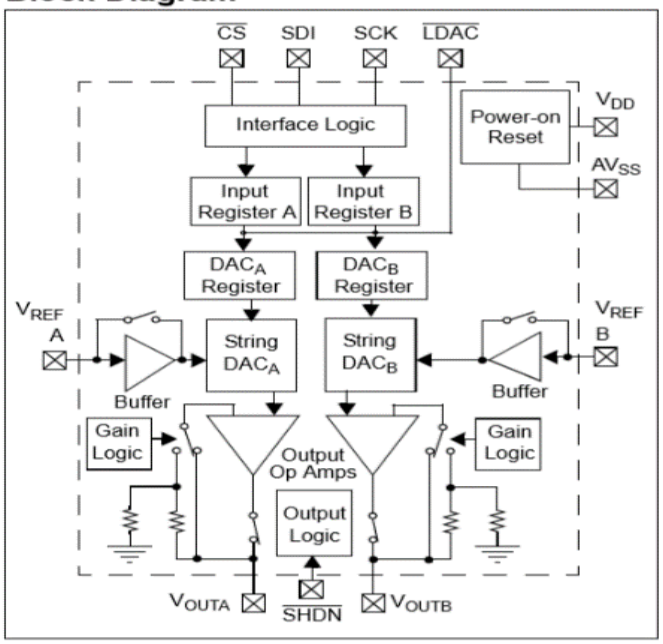


Fig 5.2.1 Block Diagram of DAC MCP4921

5V AC/DC CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 5V, AV_{SS} = 0V, V_{REF} = 2.048V, output buffer gain (G) = 2x, R_L = 5 kΩ to GND, C_L = 100 pF T_A = -40 to +85°C. Typical values at +25°C.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Power Requirements						
Input Voltage	V_{DD}	2.7	—	5.5		
Input Current - **MCP4921** - **MCP4922**	I_{DD}	— —	175 350	350 700	µA	Input unbuffered, digital inputs grounded, output unloaded, code at 0x000
Hardware Shutdown Current	I_{SHDN}	—	0.3	2	µA	
Software Shutdown Current	I_{SHDN_SW}	—	3.3	6	µA	
Power-on-Reset Threshold	V_{POR}	—	2.0	—	V	

Fig 5.2.2 Functional Characteristics of DAC

5.2.3 Pin Descriptions

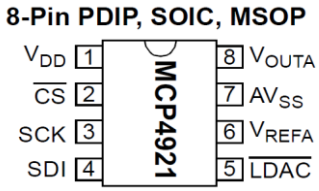


Fig 5.2.3 Pin Diagram of MCP4921

1 Positive Power Supply Input (VDD)

VDD is the positive power supply input. The input power supply is relative to AVSS and can range from 2.7V to 5.5V. A decoupling capacitor on VDD is recommended to achieve maximum performance.

2 Chip Select (CS)

CS is the chip select input, which requires an active-low signal to enable serial clock and data functions.

3 Serial Clock Input (SCK)

SCK is the SPI compatible serial clock input.

4 Serial Data Input (SDI)

SDI is the SPI compatible serial data input.

5 Latch DAC Input (LDAC)

LDAC (the latch DAC syncronization input) transfers the input latch registers to the DAC registers (output latches) when low. Can also be tied low if transfer on the rising edge of CS is desired.

6 DACx Outputs (VOUTA, VOUTB)

VOUTA and VOUTB are DAC outputs. The DAC output amplifier drives these pins with a range of AVSS to VDD.

7 DACX Voltage Reference Inputs (VREFA, VREFB)

VREFA and VREFB are DAC voltage reference inputs. The analog signal on these pins is utilized to set the reference voltage on the string DAC. The input signal can range from AVSS to VDD.

8 Analog Ground (AVSS)

AVSS is the analog ground pin.

5.2.4 Serial Interface

The MCP492X family is designed to interface directly with the Serial Peripheral Interface (SPI) port, available on many microcontrollers, and supports Mode 0,0 and Mode 1,1. Commands and data are sent to the device via the SDI pin, with data being clocked-in on the rising edge of SCK. The communications are unidirectional and, thus, data cannot be read out of the MCP492X. The CS pin must be held low for the duration of a write command. The write command consists of 16 bits and is used to configure the DAC's control and data latches. Register 5-1 details the input registers used to configure and load the DACA and DACB registers. Refer to Figure 1-1 and Section 1.0 "Electrical Characteristics" AC Electrical Characteristics table for detailed input and output timing specifications for both Mode 0,0 and Mode 1,1 operation.

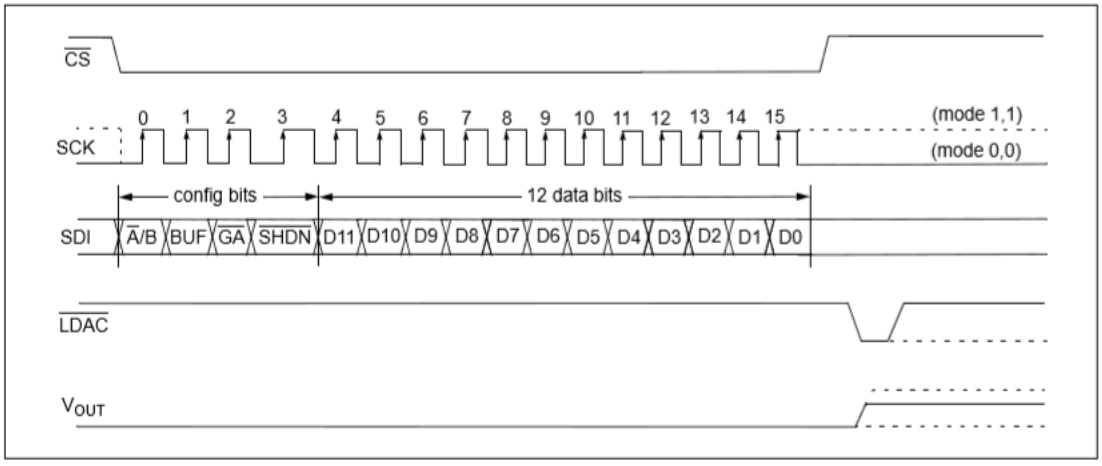


Fig 5.2.4 Timing Diagram of DAC

5.3 Field Programmable Gate Array (Fpga)

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). (Circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare.). FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

Fig 5.3.1 FPGA IC

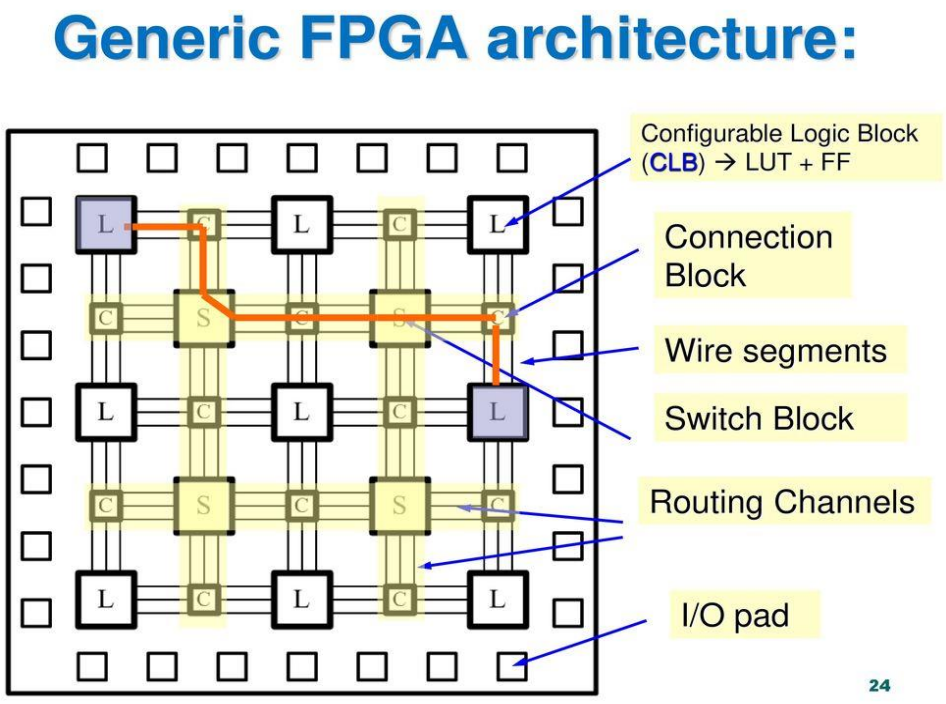


Fig 5.3.2 FPGA Architecture

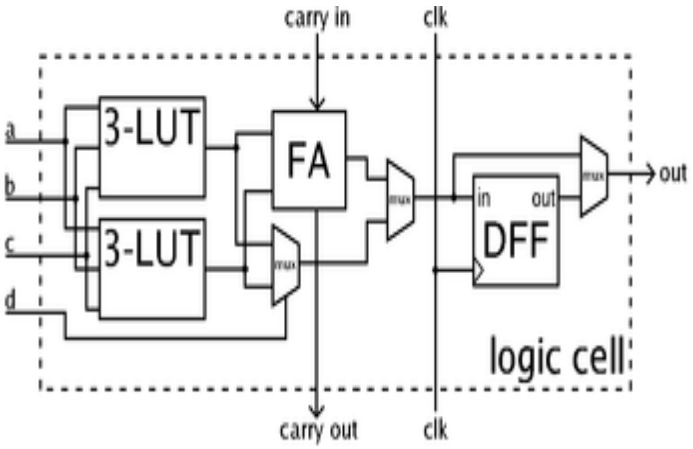


Fig 5.3.3 Logic Cells

5.4 Cost of Components

FPGA: 4800 Rs

ADC: 100 Rs

DAC: 110 Rs

6. SOFTWARE- F.I.R Filter IP Core v6.3

6.1 Functional Description Overview

A wide range of filter types can be implemented in the Xilinx CORE Generator tool: single-rate, polyphase decimators and interpolators and half-band decimators and interpolators. Structure in the coefficient set is exploited to produce area-efficient FPGA implementations. Sufficient arithmetic precision is employed in the internal datapath to avoid the possibility of overflow. The conventional single-rate FIR version of the core computes the convolution sum defined in Equation 1, where N is the number of filter coefficients. Equation 1 Figure 1 illustrates the conventional tapped delay line realization of this inner-product calculation, and although the illustration is a useful conceptualization of the computation performed by the core, the actual FPGA realization is quite different. One or more time-shared multiply-accumulate (MAC) functional units are used to service the N sum-of-product calculations in the filter. The core automatically determines the minimum number of MAC engines required to meet user-specified throughput.

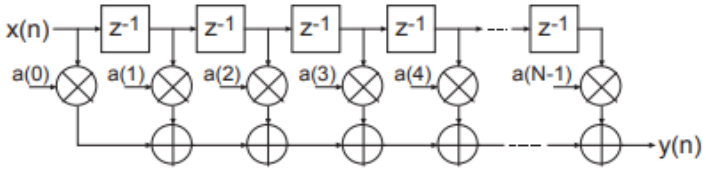


Fig 6.1 FIR Filter Structure

6.2 Pinout

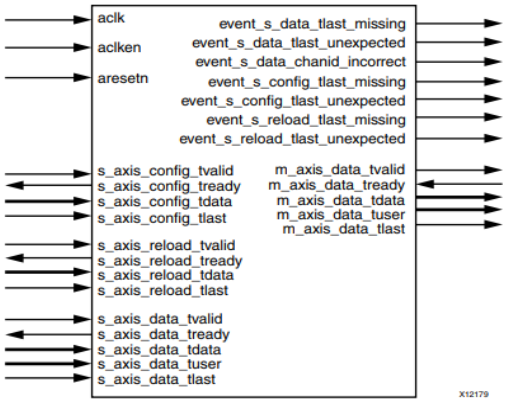


Fig 6.2 Pin Diagram of FIR IP Core

Filter input data is supplied on the s_axis_data_tdata port (N bits wide, extended if necessary to fit a byte boundary), subject to the s_axis_data_tvalid and s_axis_data_tready handshake,

and filter output samples are presented on the m_axis_data_tdata port (R bits wide, extended if necessary to fit a byte boundary), subject to the m_axis_data_tvalid and m_axis_data_tready handshake. The maximum output width R is the sum of the data bit width N and the bit growth of the filter; see the Output Width and Bit Growth section for more details. The output width can also be reduced further under user control by truncation or rounding.

s_axis_data_tvalid : TVALID for input DATA channel. Asserted by external master to indicate data is available for transfer.

s_axis_data_tready: TREADY for input DATA channel. Asserted by core to indicate core is ready to accept data.

s_axis_data_tdata: TDATA for input DATA channel. Conveys the data stream to be filtered. See TDATA Structure for internal structure.

m_axis_data_tvalid: TVALID for output DATA channel. Asserted by core to indicate data is available for transfer.

m_axis_data_tdata: TDATA for the output DATA channel. This is the filtered data stream. See TDATA Structure for internal structure.

6.3 AXI4-Stream Considerations

The conversion to AXI4-Stream interfaces brings standardization and enhances interoperability of Xilinx LogiCORE IP solutions. Other than general control signals such as aclk, aclken and aresetn and the event outputs, all inputs and outputs to the FIR Compiler are conveyed via AXI4-Stream channels. A channel consists of TVALID and TDATA always, plus several optional ports. In the FIR Compiler, the optional ports supported are TREADY, TLAST and TUSER. Together, TVALID and TREADY perform a handshake to transfer a message, where the payload is TDATA, TUSER and TLAST. The FIR Compiler operates on the data contained in the input DATA channel TDATA port (s_axis_data_tdata) and outputs the result in the TDATA field of the output DATA channel (m_axis_data_tdata).

6.4 Basic Handshake

Figure 3 shows the transfer of data in an AXI4-Stream channel. TVALID is driven by the source (master) side of the channel and TREADY is driven by the receiver (slave). TVALID indicates that the value in the payload fields (TDATA, TUSER and TLAST) is valid. TREADY indicates that the slave is ready to receive data. When both TVALID and TREADY are true in a cycle, a transfer occurs. The master and slave set TVALID and TREADY respectively for the next transfer appropriately. Some channels can be configured to have no TREADY, in which case the channel behaves as through there was an implicit, permanently asserted TREADY.

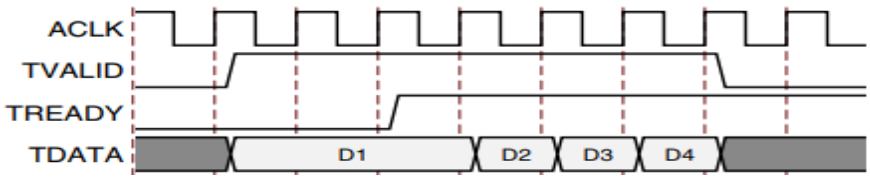


Fig 6.3 Handshake Timing Diagram

6.5 Input and Output DATA Channels

The basic operation of the FIR is for samples to enter via the input DATA channel (s_axis_data_t*) and exit via the output DATA channel (m_axis_data_t*) duly filtered. The output channel optionally supports TREADY which allows a resource/behavior trade-off. In circumstances where downstream slave can be guaranteed to accept the maximum bandwidth of the FIR, TREADY can be deselected to save resources. The input DATA channel always supports TREADY

TREADY and TVALID- All AXI4-Stream channels support TVALID. The input DATA channel also always supports TREADY. The output channel optionally supports TREADY. Back-pressure from the output channel eventually propagates to the input DATA channel to ensure that no data is dropped.

6.6 Systolic Multiply-Accumulate

Figure 4 illustrates the Systolic Multiply-Accumulate architecture implementing a pipelined Direct-Form filter.

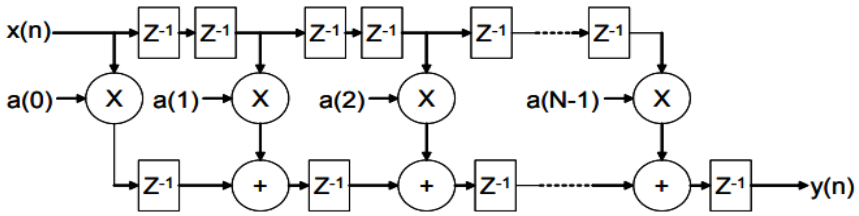


Fig 6.4 Systolic Multiply- Accumulate Structure

Single-rate FIR Filter: The basic FIR filter core is a single-rate (input sample rate = output sample rate) finite impulse response filter. This is the simplest of filter types and is the default at the start of parametrization in the CORE Generator software.

6.7 MAC-based FIR Filter Timing (Single Rate)

Figure 5 illustrates the timing for a single-rate, single-channel, N-tap MAC-based filter. ND(**s_axis_data_tvalid**) is asserted while valid input is available on the DIN(**s_axis_data_tdata**) port. At the rising edge of the clock, the data is sampled and processing begins. RFD(**s_axis_data_tready**) is deasserted to reflect that the MAC-based FIR core is processing the data and unable to accept further input samples for the period of the input data rate. After a number of clock cycles equal to the "Cycle Latency," RDY(**m_axis_data_tvalid**) is asserted and the valid filter output is presented on the DOUT(**m_axis_data_tdata**) port. In this example, the DOUT value is held in the optional output register. In this configuration, core operation can be halted by holding ND(**s_axis_data_tvalid**) low for the required idle note. However, the core continues to process any input data sampled so far and to produce outputs based on those input samples.

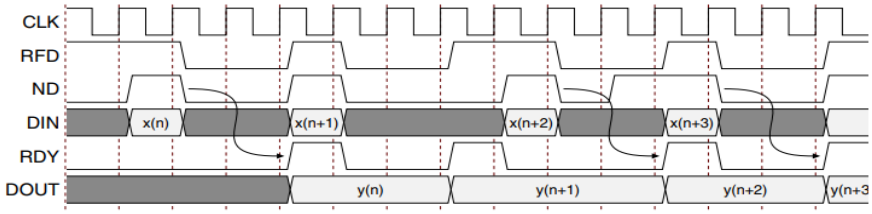


Fig 6.5 FIR Timing Diagram

NOTE:

CLK=aclk, ND= s_axis_data_tvalid, RFD=s_axis_data_tready, RDY=m_axis_data_tvalid, DOUT= (m_axis_data_tdata), DIN= (s_axis_data_tdata).

7. RESULTS

7.1 Low Pass Filter Results:

Filter coefficient set corresponding to 21st order low pass FIR filter was uploaded in the FIR Xilinx IP Core. Using MATLAB the frequency response corresponding to the coefficient set was plotted and is shown in the fig 1. After implementing the Filter in the FPGA the practical frequency response is plotted in fig2.

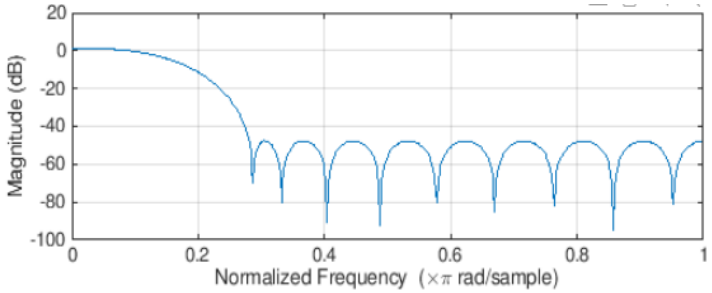


Fig 7.1 Ideal Frequency Response LPF

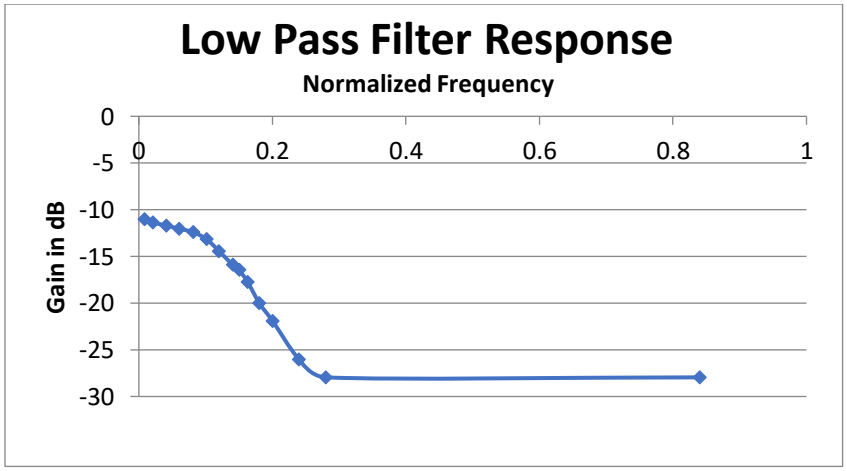


Fig 7.2 Observed Frequency Response LPF

Result: Ideal Cut-off Frequency (-3dB): 0.14; Practical Cut-off Frequency (-3dB): 0.12

7.2 High Pass Filter Results:

Filter coefficient set corresponding to 41st order High pass FIR filter was uploaded in the FIR Xilinx IP Core. Using MATLAB the frequency response corresponding to the coefficient set was plotted and is shown in the fig 1. After implementing the Filter in the FPGA the practical frequency response is plotted in fig2.

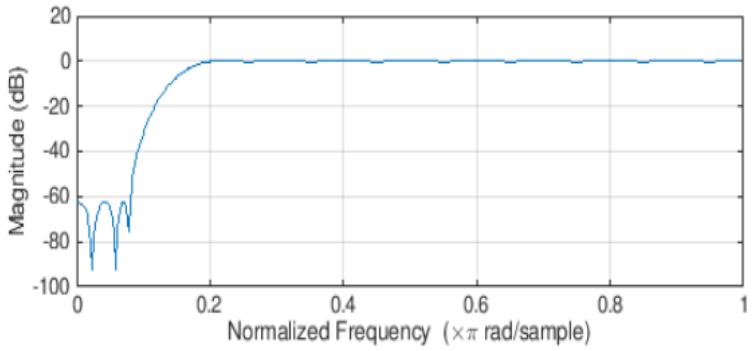


Fig 7.3 Ideal Frequency Response HPF

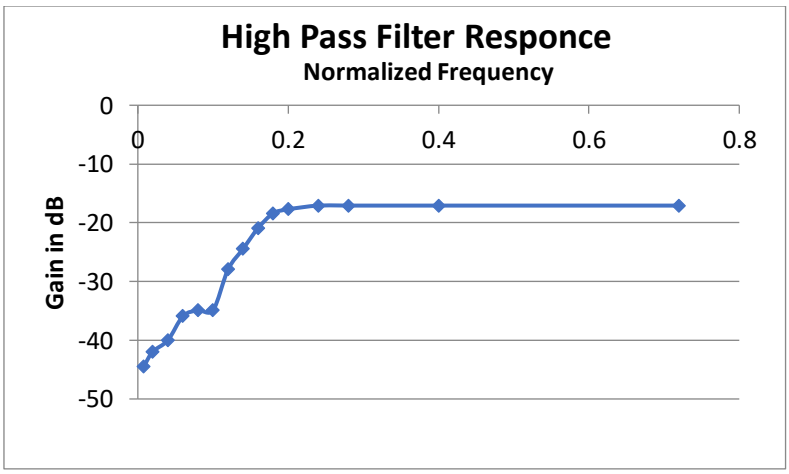


Fig 7.4 Observed Frequency Response

Result: Ideal Cut-off Frequency (-3dB): 0.1699; Practical Cut-off Frequency (-3dB): 0.16

7.3 **Band Pass Filter Results** :

Filter coefficient set corresponding to 41st order low pass FIR filter was uploaded in the FIR Xilinx IP Core. Using MATLAB the frequency response corresponding to the coefficient set was plotted and is shown in the fig 1. After implementing the Filter in the FPGA the practical frequency response is plotted in fig2.

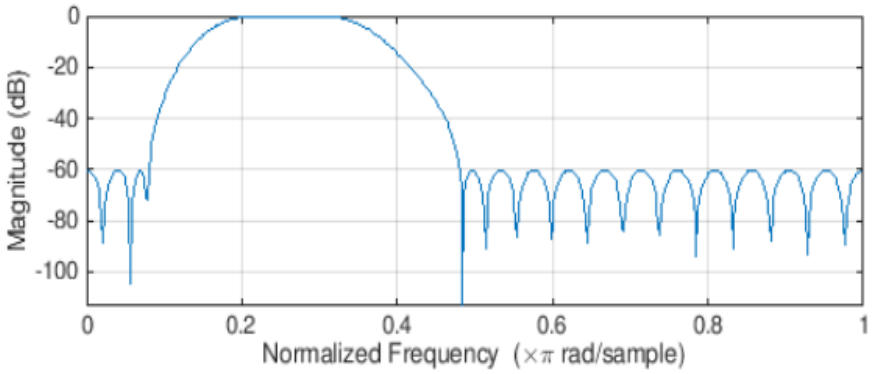


Fig 7.5 Ideal Frequency Response BPF

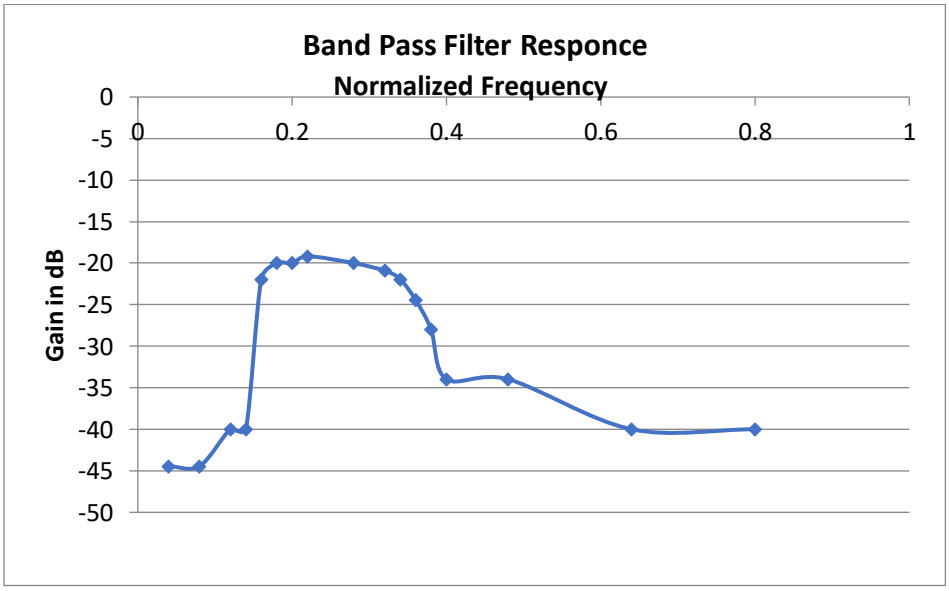


Fig 7.6 Observed Frequency Response BPF

Result: Ideal cut-off frequencies fc1= 0.1699, fc2= 0.3496.
Practical Cut-off Frequencies fc1= 0.16, fc2 =0.34.

7.4 ADC and DAC Simulation Results.

Here it can be seen that in the simulation 0 volts is given as an input to the ADC (assumption) and hence the ADC_MISO pin is always Low. Here the ADC protocol and DAC protocol is successfully implemented and it can be verified with the Timing Diagram given in the Datasheet. ADC Timing Diagram includes ADC_SCK, ADC_MOSI, ADC_CS_BAR and ADC_MISO. DAC Timing Diagram includes DAC_SCK, DAC_MOSI, DAC_CS_BAR. Because of this Serial Peripheral Interface (SPI) ADC IC MCP3202 is successfully able to sample the signal at 50ksps and can be sent on the Data Bus. And the DAC MCP4921 is also able to successfully able to convert the data from digital to analog domain.

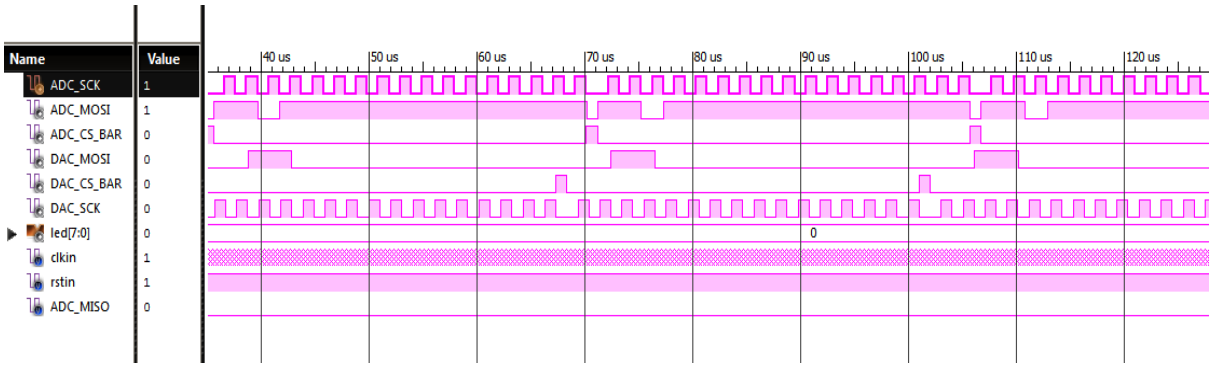


Fig 7.7 ADC and DAC Simulation

7.5 DAC Simulation Results

Here the pins involved in implementing Serial Peripheral Interface (SPI) Protocol are MOSI, CS_bar, SCK and digital_data. 'digital_data [11:0]' is given as a parallel input to DAC Module which produces the Serial Output 'MOSI' which is then passed to the DAC IC MCP4921. The simulation Results matches exactly with the timing diagram given in the datasheet.

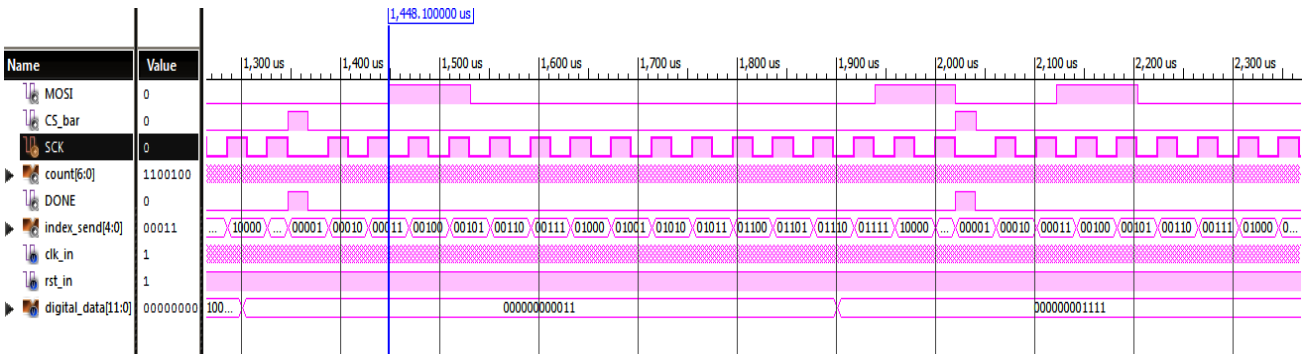


Fig 7.8 DAC Simulation

8. Problems Faced

1. The FIR filter wasn't giving the Frequency response as expected.

After uploading the Filter Coefficients it was observed though the filter was giving perfect Step Response and Impulse Response when seen in the simulation (Isim), it wasn't filtering the sine wave of all the frequencies which were given externally from the Function Generator.

Solution:

To solve this issue we again analyzed our design. It was seen that the clock latency of our filter was 19 clock pulses. That means the filter was ready after 19clock cycle to accept the new data. But we were providing the new data after 19 clock cycle to the filter to produce the corresponding data. But since our ADC sampling frequency was 50ksps hence new sampled data pulse will enter the FPGA(system) after 20usec(2000 clock pulse). But initially we were allowing the filter to accept the data after 19 clock pulses (i.e after 0.19usec). That means Filter was accepting the same data again and again till the actual new sampled data comes inside the FPGA. Which is not advisable filter is supposed to accept the new data every-time (i.ex(n),x(n+1), x (n+2) ....). Hence it was our responsibility to send the data after 2000 clock pulse. By doing this the FIR filter successful filtered the signal in Real time which was given from the function generator.

2. FIR filter was not giving any output in the simulation.

After uploading the Filter Coefficients it was observed that the filter was not giving any output in the simulation. This was because the filter was not accepting any data. From the data sheet it was written that input data will be sampled at rising edge of the clock, when the s_axis_data_tvalid is asserted, as shown in the fig.1. But this did not work.

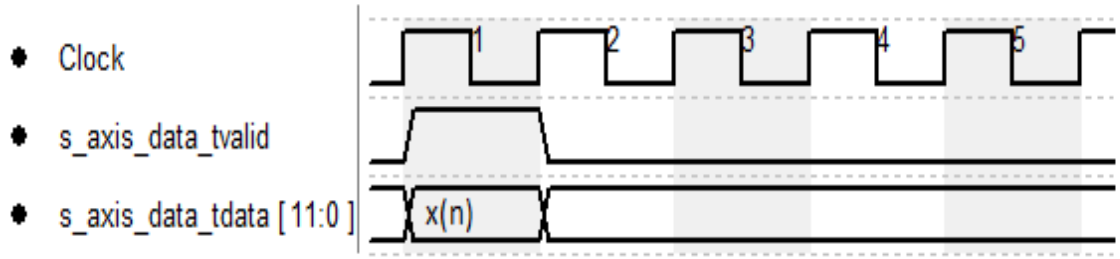


Fig 8.1 Timing Diagram FIR IP Core (not working)

Solution:

By experimenting the timing diagram shown below (fig.2) it was observed that the input signal was successfully sampled at the positive edge trigger of the clock and when 's_axis_data_tvalid' was asserted.

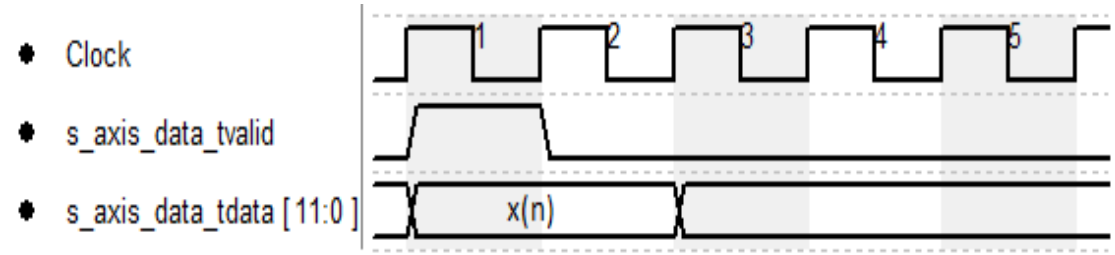


Fig 8.2 Timing Diagram FIR IP Core (working)

9. Limitations

The limitations of the project are as follows:

1. The inbuilt DC gain of the Filter is 0.018. H(0)=0.018
2. The sampling frequency of the ADC is 50ksps. This is because of the clock frequency which is kept at 0.5MHz and the power supply provided to the ADC MCP3202 is 3.3V.
3. The signal should not have frequency greater than 25KHz. This is because of the Nyquist Criteria which says that the signal frequency should be less than half of the Sampling Frequency.
4. The Design only allows us to implement Finite Impulse Response (FIR) Low Pass, Band Pass and High Pass Filter.

10. Conclusion

Thus in this project, the implementation of an FIR Filter using FPGA was successfully carried out. Using this project, we can now process and filter Real Time Signals from the function generator and observe the output in real time on the CRO. The filter can behave as any type of filter as required, be it Low Pass Filter, High Pass Filter, Band Pass Filter or Band Stop Filter. The Gain or Attenuation of the filter can also be controlled.

All this is achieved simply by loading the appropriate Filter Coefficients.

11. Future scope

The Project can be further improved in the future by using the ADC and DAC which provide better Frequency Sampling Rate and better Resolution. Also, the FPGA can be further programmed to accept a bunch of coefficients which can be permanently stored and the user can change the system into the required filter on the go. This will make the system Dynamically Reconfigurable.

12. Technical Paper

REAL TIME SIGNAL PROCESSING USING FPGA

Gurkiran Singh Bhatia
Student, EXTC
Thadomal Shahani Engineering College
Mumbai, India

Rohan Sudhir Nalavade
Student, EXTC
Thadomal Shahani Engineering College
Mumbai, India

Harsh Kamlesh Rathore
Student, EXTC
Thadomal Shahani Engineering College
Mumbai, India

Ms. Neeru Pathak
Assistant Professor, EXTC
Thadomal Shahani Engineering College
Mumbai, India

***Abstract* - This paper discusses the objectives of a design of FIR filters on Field Programmable Gate Array (FPGAs) using IP core. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, implementation of fixed-point digital signal processing algorithms, reconfigurability, and more flexibility, than the alternate approaches.**

Keywords – Field Programmable Gate Array, Digital Signal Processing, Finite Impulse Response, Filters, ADC, DAC;

I. INTRODUCTION

Recent advances in FPGA technology have enabled these devices to be applied to a variety of applications traditionally reserved for ASICs. FPGAs are well suited to datapath designs, such as those encountered in digital filtering applications. The density of the new programmable devices is such that a nontrivial number of arithmetic operations such as those encountered in digital filtering may be implemented on a single device. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches. In particular, multiple multiply-accumulate (MAC) units may be implemented on a single FPGA, which provides comparable performance to general-purpose architectures which have a single MAC unit. Further, since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement alternate filtering operations, such as lattice filters and gradient-based adaptive filters, or entirely different.

FPGAs are well suited for the implementation of fixed-point digital signal processing algorithms. The advantages of DSP on FPGAs are primarily related to the additional flexibility provided by FPGA reconfigurability. Not only can high-performance systems be implemented relatively inexpensively, but the design and test cycle can be completed rapidly due to the elimination of the integrated circuit fabrication delays. The new approach also allows adapting the functions to account for unforeseen requirements.

In binary arithmetic, multiplication by a power-of-two is simply a shift operation. Implementation of systems with multiplications may be simplified by using only a limited number of power-of-two terms, so that only a small number of shift and add operations are required. An FIR filter tap as shown in can be implemented in two array columns of Xilinx series FPGAs. Because of the high degree of spatial and temporal locality, most signal routing delays are not critical, as they are with typical high performance FPGA designs. Each of the bit slices for the tap requires two combinational logic blocks (CLBs) in the array for implementation.

This board has the following features, relevant to the presented implementation :

1) Spartan6 FPGA with 9152 logic cells (XC6SLX9),

2) 100MHz crystal oscillator,

3) Expansion connector with 32 I/O pins,

The scheme used here for the implementation of FIR filter into the FPGA is MAC based. The main components of the implemented circuit are as follows:

A. Memory: Prepared for storage of past position data of bunches. z-1 means 1-turn delay .

B. Adder: Adder is made to reduce the number of stages and is a key for stable operation of the FPGA used in the board. This reduction of the stage is effective to avoid errors by clock skew in the FPGA and to reduce the power consumption and a circuit area(or number of gates) on FPGA.

C. Multiplier: Build-in multipliers are used to fulfil the requirements of high-speed operation; therefore this number of build-in multipliers is one of the constraints to the number of taps of FIR filter.

D. Shift-Register: It is used for additional delay for adjust latency to one or two revolution period.

II. LITERATURE REVIEW

A Filter is frequency selective network, which is used to modify an input signal in order to facilitate further processing. Basically there are two types of filters-analog and digital. Digital Filters are widely used in different areas, because Digital filters have the potential to attain much better signal to noise ratio than analog filters. The digital filter performs noiseless mathematical operations at each intermediate step in the transform and their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters.

Digital filters operate on numbers opposite to analog filters, which operates on voltages. The basic operation of digital filter is to take a sequence of input numbers and compute a different sequence of output numbers. There exists a range of different digital filters. FIR and IIR filters are the two common filter forms. A drawback of IIR filters is that the closed-form IIR designs are preliminary limited to low pass, band pass, and high pass filters, etc. secondly FIR filters can have precise linear phase. Also, in the case of FIR filters, closed-form design equations do not exist and the design problem for FIR filters is much more under control than the IIR design problem. A FIR filter is a filter structure that can be used to implement almost any sort of frequency response digitally. It is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. The architecture of FIR filter is shown in Fig 1.

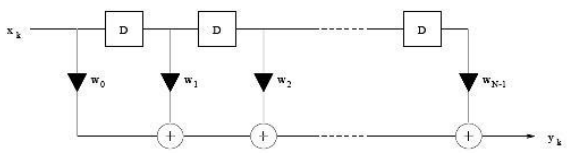


Fig.1: Basic FIR Filter Structure

From this structure the transfer function of canonic form of the filter can be easily described in Z-domain as:

$H(Z)=w_0+w_1z^{-1}+w_2z^{-2}+------+w_{M-1}Z^{L-1}$

In addition the advantages of FIR filter generally we use raised cosine pulse. The rectangular pulse occupies a large bandwidth so an alternative to rectangular pulse is a sinc pulse, which reduces the bandwidth and Inter Symbol Interference. The rectangular pulse is passed through the Root Cosine Filter a set of FIR filters to pulse shape the pulses to sinc. If very high sampling rates are required, full parallel hardware must be used. Such filters can be implemented on FPGAS using combinations of the general-purpose logic fabric, on-board RAM and embedded arithmetic hardware. Full-parallel filters cannot share hardware over multiple clock cycles and so tend to occupy large amounts of resource. Hence, efficient implementation of such filters is important to minimize hardware requirement.

III. PROPOSED MODEL

The aim of the project involves the development of FIR filters on Field programmable gate array (FPGAs) using IP cores.

FIR filter has been designed and realized by FPGA for filtering the digital signal. The implementation of FIR filter on a Xilinx XC65LX9 FPGA is considered and the coefficients are computed using Online FDA tool. The model is capable of performing filtering operations like low pass, high pass, and band pass based on the coefficients selected. The most basic functions required for nearly any signal processor include addition, multiplication and delays. The analog input given is converted to a 12 bit digital data. IP Core has been used to filter the input data. The design is coded through Verilog (Hardware Descriptive Language). To verify the designed outputs simulation, compilation and synthesis have been done. To test the correctness of the design the observed output is compared with the calculated output results from MATLAB implementation that confirms the effectiveness of the design.

Computer-aided design tools are used to both simulate VERILOG design and to synthesize the design to actual hardware. The Xilinx Floating-Point core is a function inbuilt in IP cores provides designers with the means to perform floating-point arithmetic on an FPGA. The core

can be customized to allow optimization for operation, word length, latency, and interface

The designing of an FIR filter in VERILOG with MATLAB (for the generation of coefficients of filter) and programming it onto an FPGA is done.

IV. DESIGN AND IMPLEMENTATION

Block Diagram of the System which is implemented inside the FPGA is given in the Fig.2.

Working of the module is as follow.The 'clk_sys' is the system clock of the FPGA given to the four blocks.Spi-master1 communicates with the external ADC.The data given to the ADC from the function generator is converted into a digital signal. This is then given to the FPGA. In the FPGA, the module spi_master1 will accept the input digital signal in a serial manner. The spi_master1 will then convert the serial input digital signal into a 12 bit parallel data referred as RDATA inside the spi_master1.

This data is then transferred to the fir_ipcore and is referred to as ADC_DATA. The fir_ipcore will accept the data and will do the filtering depending upon what kind of coefficients are loaded into the IP Core.

The fir_ipcore accepts the data when it is not processing any data. The decision of when the fir_ipcore should accept the data is made by the drive_fir1 block. The drive_fir1 block knows the rate at which the samples are coming from the spi_master1 and accordingly sends the data_tvalid signal which indicates to the fir_ipcore that a valid input has now come at s_tdata.

After processing the signal, the fir_ipcore sends the data to spi_dac1. The data is referred to as digital_data in the spi_dac1. Here the data is converted into serial form. spi_dac1 communicates with the external DAC. It transmits the serial data referred as DAC_MOSI. This data goes to the external Digital to Analog Converter (DAC) which converts this digital data into analog signal. This analog signal is then observed on the Cathode Ray Oscilloscope (CRO).

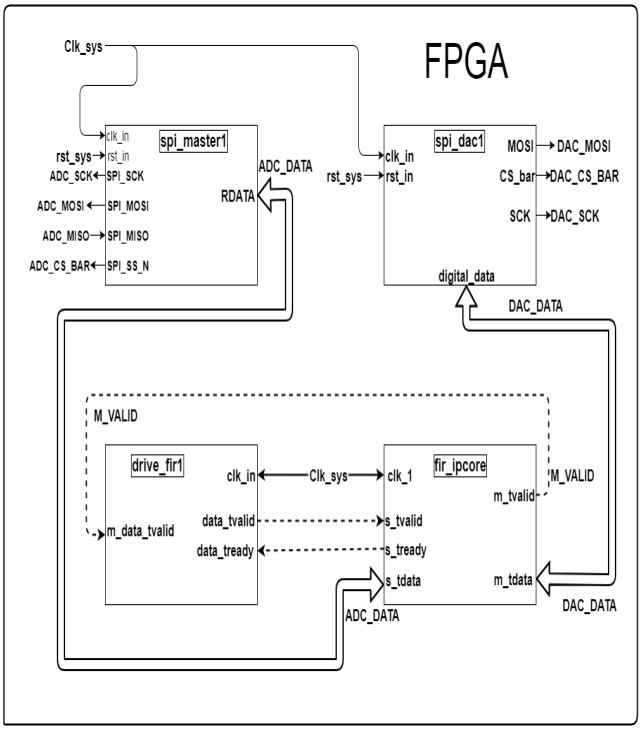


Fig.2: Block Diagram of the System

V. COMPONENTS USED

A. *Analog to Digital Converter MCP3202*

- 12-bit resolution
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V-5.5V
- 100 ksps max. sampling rate at VDD = 5V
- 50 ksps max. sampling rate at VDD = 2.7V
- Low power CMOS technology - 500 nA typical standby current, 5 µA max. - 550 µA max. active current at 5V
- Industrial temp range: -40°C to +85°C
- 8-pin MSOP, PDIP, SOIC and TSSOP packages

Pin Description:

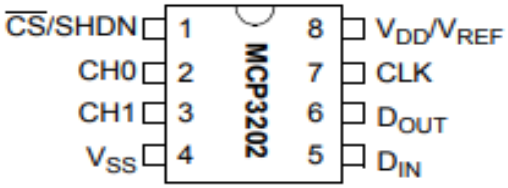


Fig.3: Pin Description of MCP3202

1. CH0/CH1:

Analog inputs for channels 0 and 1 respectively. These channels can programmed to be used as two independent channels in single ended-mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN-.

2. Chip Select/Shutdown (CS/SHDN):

The CS/SHDN pin is used to initiate communication with the device when pulled low and will end a conversion and put the device in low power standby when pulled high. The CS/SHDN pin must be pulled high between conversions.

3. Serial Clock (CLK):

The SPI clock pin is used to initiate a conversion and to clock out each bit of the conversion as it takes place.

4. Serial Data Input (DIN):

The SPI port serial data input pin is used to clock in input channel configuration data.

5. Serial Data Output (DOUT):

The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

B. Digital to Analog Converter MCP4921

- 12-Bit Resolution
- ±0.2 LSB DNL (typ)
- ±2 LSB INL (typ)
- Single or Dual Channel
- SPI™ Interface with 20 MHz Clock Support
- Fast Settling Time of 4.5 µs
- Selectable Unity or 2x Gain Output
- 450 kHz Multiplier Mode
- External VREF Input
- 2.7V to 5.5V Single-Supply Operation

Pin Description:

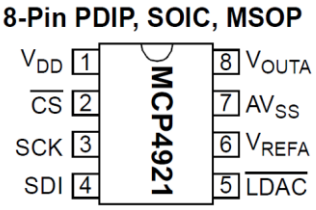


Fig.4: Pin Diagram of MCP4921

1. Positive Power Supply Input (VDD)

VDD is the positive power supply input. The input power supply is relative to AVSS and can range from 2.7V to 5.5V. A decoupling capacitor on VDD is recommended to achieve maximum performance.

2. Chip Select (CS)

CS is the chip select input, which requires an active-low signal to enable serial clock and data functions.

3. Serial Clock Input (SCK)

SCK is the SPI compatible serial clock input.

4. Serial Data Input (SDI)

SDI is the SPI compatible serial data input.

5. Latch DAC Input (LDAC)

LDAC (the latch DAC synchronization input) transfers the input latch registers to the DAC registers (output latches) when low. Can also be tied low if transfer on the rising edge of CS is desired.

6. DACx Outputs (VOUTA, VOUTB)

VOUTA and VOUTB are DAC outputs. The DAC output amplifier drives these pins with a range of AVSS to VDD.

7. DACX Voltage Reference Inputs (VREFA, VREFB)

VREFA and VREFB are DAC voltage reference inputs. The analog signal on these pins is utilized to set the reference voltage on the string DAC. The input signal can range from AVSS to VDD.

8. Analog Ground (AVSS)

AVSS is the analog ground pin.

C. Field Programmable Gate Array (FPGA)

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). (Circuit diagrams were previously

used to specify the configuration, as they were for ASICs, but this is increasingly rare.). FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

Fig.5: FPGA IC

VI. SOFTWARE- XILINX FIR IP CORE V6.3.

Pin Diagram:

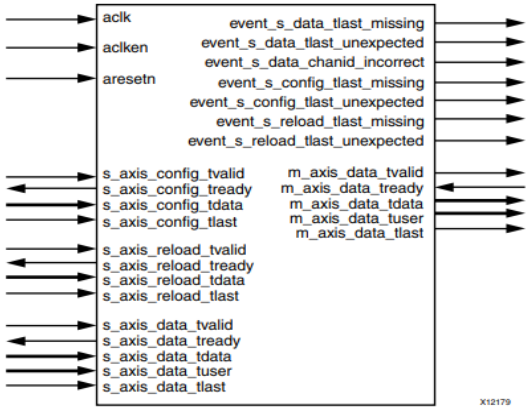


Fig.6: Pin Diagram of FIR IP Core

Filter input data is supplied on the s_axis_data_tdata port (N bits wide, extended if necessary to fit a byte boundary), subject to the s_axis_data_tvalid and s_axis_data_tready handshake, and filter output samples are presented on the m_axis_data_tdata port (R bits wide, extended if necessary to fit a byte boundary), subject to the m_axis_data_tvalid and m_axis_data_tready handshake. The maximum output width R is the sum of the data bit width N and the bit growth of the filter; see the Output Width and Bit Growth section for more details. The output width can also be reduced further under user control by truncation or rounding.

's_axis_data_tvalid': TVALID for input DATA channel. Asserted by external master to indicate data is available for transfer.

's_axis_data_tready': TREADY for input DATA channel. Asserted by core to indicate core is ready to accept data.

's_axis_data_tdata': TDATA for input DATA channel. Conveys the data stream to be filtered. See TDATA Structure for internal structure.

'm_axis_data_tvalid': TVALID for output DATA channel. Asserted by core to indicate data is available for transfer.

'm_axis_data_tdata': TDATA for the output DATA channel. This is the filtered data stream. See TDATA Structure for internal structure.

VII. RESULTS AND ANALYSIS

A. Low Pass Filter Results:

Filter coefficient set corresponding to 21st order low pass FIR filter was uploaded in the FIR Xilinx IP Core. Using MATLAB the frequency response corresponding to the coefficient set was plotted and is shown in the fig 1. After implementing the Filter in the FPGA the practical frequency response is plotted in fig2.

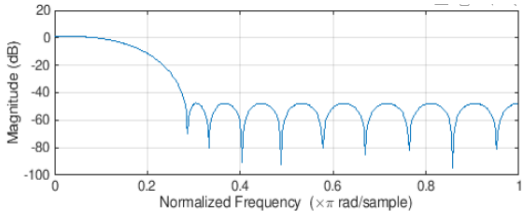


Fig.7: Ideal Frequency Response LPF

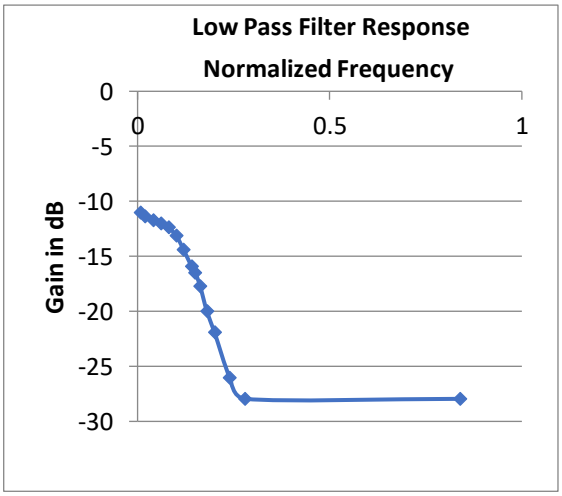


Fig.8: Observed Frequency Response LPF

Result: Ideal Cut-off Frequency (-3dB): 0.14; Practical Cut-off Frequency (-3dB): 0.12

B. High Pass Filter Results:

Filter coefficient set corresponding to 41st order High pass FIR filter was uploaded in the FIR Xilinx IP Core. Using MATLAB the frequency response corresponding to the coefficient set was plotted and is shown in the fig 1. After implementing the Filter in the FPGA the practical frequency response is plotted in fig2.

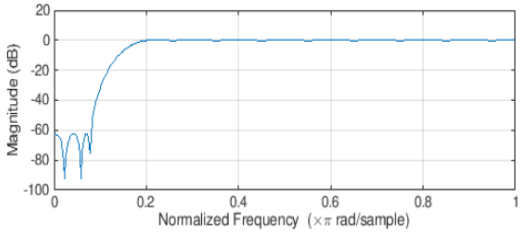


Fig.9: Ideal Frequency Response HPF

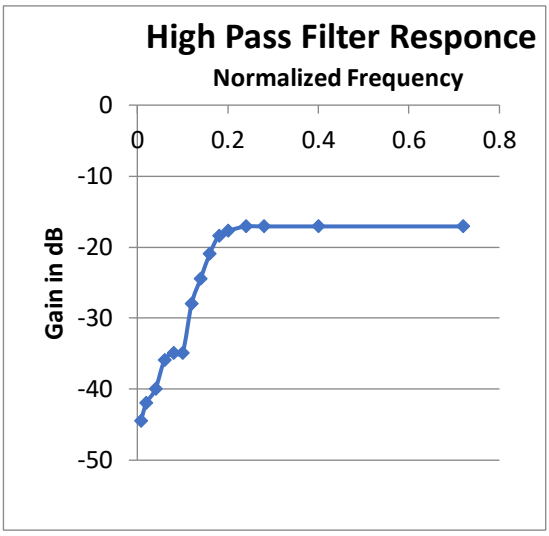


Fig.10: Observed Frequency Response

Result: Ideal Cut-off Frequency (-3dB): 0.1699; Practical Cut-off Frequency (-3dB): 0.16

C. Band Pass Filter Results :

Filter coefficient set corresponding to 41st order low pass FIR filter was uploaded in the FIR Xilinx IP Core. Using MATLAB the frequency response corresponding to the coefficient set was plotted and is shown in the fig 1. After implementing the Filter in the FPGA the practical frequency response is plotted in fig2.

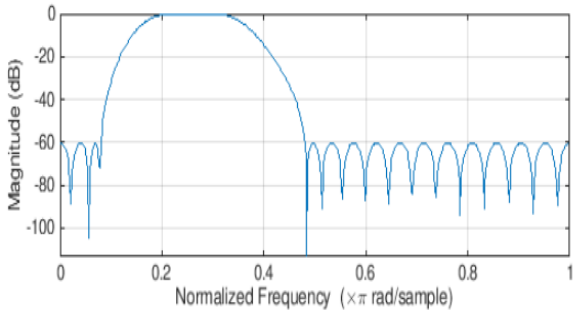


Fig.11: Ideal Frequency Response BPF

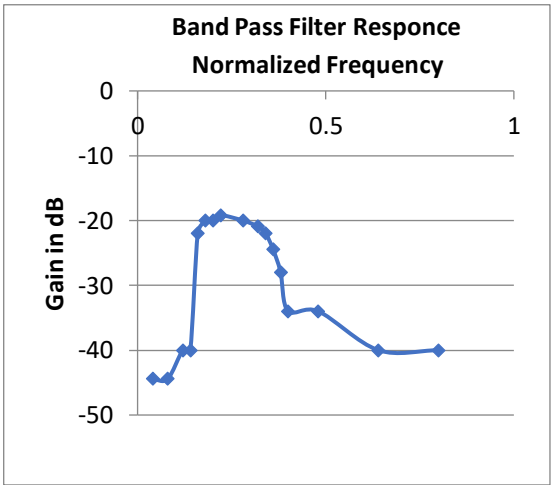


Fig.12: Observed Frequency Response BPF

Result: Ideal cut-off frequencies fc1= 0.1699, fc2= 0.3496.

Practical Cut-off Frequencies fc1= 0.16, fc2 =0.34.

D. ADC and DAC Simulation Results.

Here it can be seen that in the simulation 0 volts is given as an input to the ADC (assumption) and hence the ADC_MISO pin is always Low. Here the ADC protocol and DAC protocol is successfully implemented and it can be verified with the Timing Diagram given in the Datasheet. ADC Timing Diagram includes ADC_SCK, ADC_MOSI, ADC_CS_BAR and ADC_MISO. DAC Timing Diagram includes DAC_SCK, DAC_MOSI, DAC_CS_BAR. Because of this Serial Peripheral Interface (SPI) ADC IC MCP3202 is successfully able to sample the signal at 50ksps and can be sent on the Data Bus. And the DAC MCP4921 is also able to

successfully able to convert the data from digital to analog domain.

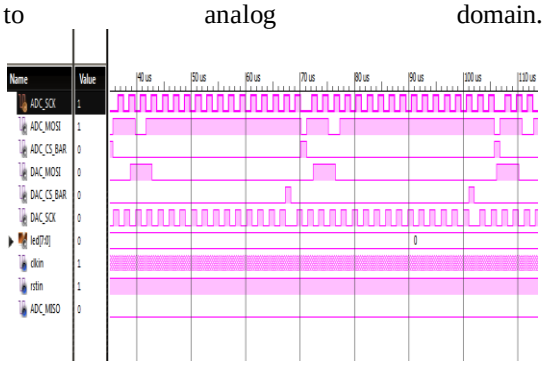


Fig.13: ADC and DAC Simulation

E. DAC Simulation Results

Here the pins involved in implementing Serial Peripheral Interface (SPI) Protocol are MOSI, CS_bar, SCK and digital_data. 'digital_data [11:0]' is given as a parallel input to DAC Module which produces the Serial Output 'MOSI' which is then passed to the DAC IC MCP4921. The simulation Results matches exactly with the timing diagram given in the datasheet.

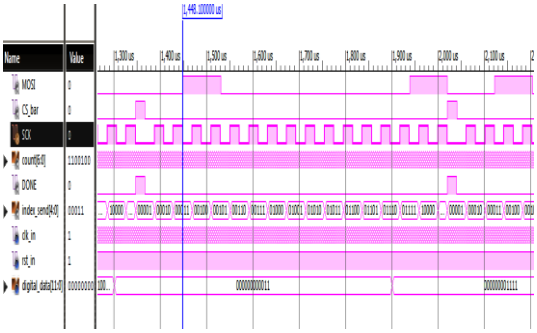


Fig.14: DAC Simulation

VIII. PROBLEMS FACED

A. The FIR filter wasn't giving the Frequency response as expected.

After uploading the Filter Coefficients it was observed though the filter was giving perfect Step Response and Impulse Response when seen in the simulation (Isim), it wasn't filtering the sine wave of all the frequencies which were given externally from the Function Generator.

Solution:

To solve this issue we again analyzed our design. It was seen that the clock latency of our filter was 19 clock pulses. That means the filter was ready after 19clock cycle to accept the new data. But we were providing the new data after 19 clock cycle to the filter to produce the corresponding data. But since our ADC sampling frequency was 50ksps hence new sampled data pulse will enter the FPGA(system) after 20usec(2000 clock pulse). But initially we were allowing the filter to accept the data after 19 clock pulses (i.e after 0.19usec). That means Filter was accepting the same data again and again till the actual new sampled data comes inside the FPGA. Which is not advisable filter is supposed to accept the new data every-time (i.ex(n),x(n+1), x (n+2) ....). Hence it was our responsibility to send the data after 2000 clock pulse. By doing this the FIR filter successful filtered the signal in Real time which was given from the function generator.

B. FIR filter was not giving any output in the simulation.

After uploading the Filter Coefficients it was observed that the filter was not giving any output in the simulation. This was because the filter was not accepting any data. From the data sheet it was written that input data will be sampled at rising edge of the clock, when the s_axis_data_tvalid is asserted, as shown in the fig.1. But this did not work.

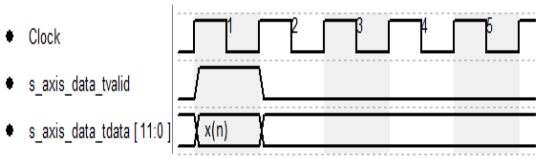


Fig 8.1 Timing Diagram FIR IP Core (not working)

Solution:

By experimenting the timing diagram shown below (fig.2) it was observed that the input signal was successfully sampled at the positive edge trigger of the clock and when 's_axis_data_tvalid' was asserted.

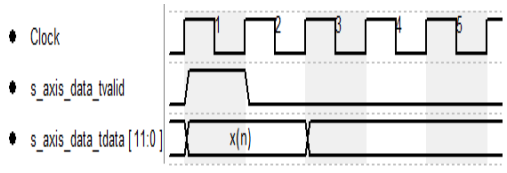


Fig 8.2 Timing Diagram FIR IP Core (working)

IX. CONCLUSION AND FUTURE SCOPE

Thus in this project, the implementation of an FIR Filter using FPGA was successfully carried out. Using this project, we can now process and filter Real Time Signals from the function generator and observe the output in real time on the CRO. The filter can behave as any type of filter as required, be it Low Pass Filter, High Pass Filter, Band Pass Filter or Band Stop Filter. The Gain or Attenuation of the filter can also be controlled.

All this is achieved simply by loading the appropriate Filter Coefficients.

The Project can be further improved in the future by using the ADC and DAC which provide better Frequency Sampling Rate and better Resolution. Also, the FPGA can be further programmed to accept a bunch of coefficients which can be permanently stored and the user can change the system into the required filter on the go. This will make the system Dynamically Reconfigurable.

REFERENCES

[1] Verilog HDL: A Guide to Digital Design and Synthesis, Second Edition By Samir Palnitkar

[2] https://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v6_3/ds795_fir_compiler.pdf

[3] http://www.farnell.com/datasheets/1669376.pdf

[4] http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf

[5] https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

13. REFERENCES

[1] Verilog HDL: A Guide to Digital Design and Synthesis, Second Edition By Samir Palnitkar

[2] https://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v6_3/ds795_fir_compiler.pdf

[3] http://www.farnell.com/datasheets/1669376.pdf

[4] http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf

[5] https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

14. Appendix

HDL Synthesis Report:Macro Statistics

Adders/Subtractors	: 11
1-bit adder	: 1
12-bit adder	: 1
14-bit adder	: 1
2-bit subtractor	: 1
4-bit adder	: 2
4-bit subtractor	: 2
5-bit adder	: 1
7-bit subtractor	: 2
Registers	: 29
1-bit register	: 20
12-bit register	: 2
16-bit register	: 1
4-bit register	: 3
5-bit register	: 1
7-bit register	: 2
Comparators	: 3
4-bit comparator greater	: 2
5-bit comparator greater	: 1
Multiplexers	: 21
1-bit 16-to-1 multiplexer	: 1
1-bit 2-to-1 multiplexer	: 12
1-bit 4-to-1 multiplexer	: 1
12-bit 2-to-1 multiplexer	: 1
16-bit 2-to-1 multiplexer	: 1
4-bit 2-to-1 multiplexer	: 2
5-bit 2-to-1 multiplexer	: 1
7-bit 2-to-1 multiplexer	: 2
FSMs	: 3

Advanced HDL Synthesis Report

Macro Statistics:

Adders/Subtractors	: 6
1-bit adder	: 1
12-bit adder	: 1
2-bit subtractor	: 1
4-bit subtractor	: 2
7-bit subtractor	: 1
Counters	: 5
12-bit up counter	: 1
4-bit up counter	: 2
5-bit up counter	: 1
7-bit down counter	: 1
Registers	: 58
Flip-Flops	: 58
Comparators	: 3
4-bit comparator greater	: 2
5-bit comparator greater	: 1
Multiplexers	: 31
1-bit 16-to-1 multiplexer	: 1
1-bit 2-to-1 multiplexer	: 28
1-bit 4-to-1 multiplexer	: 1
7-bit 2-to-1 multiplexer	: 1
FSMs	: 3